Data Analysis I/O at the Exascale

Christopher Mitchell & Jun Wang
University of Central Florida
John Patchett & James Ahrens
Los Alamos National Laboratory
Ross Miller & Galen Shipman
Oak Ridge National Laboratory

Overview

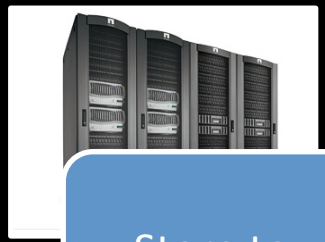
- Current Challenges with Analysis and Visualization
- Solution: *VisIO*
  Rethinking I/O architectures
- Solution: *Climate Visualization with POP*
  Rethinking I/O access patterns
- Conclusions

Current Workflows and Challenges

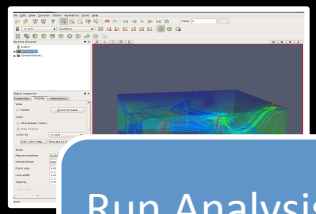
ANALYSIS AND VISUALIZATION TODAY

Today's Supercomputing Environment

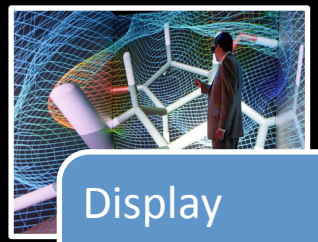
Acquire Data
- Simulation
- Experiment



Store to Parallel File System



Run Analysis & Visualization Tools



Display
- Desktop
- CAVE
- PowerWall

Simple…. Right?

Today's Challenges at the Petascale

- FLOPS FIRST!
  - Designers: Computers designed to maximize FLOPS rating.
  - Project Management: ~10% of budget (with variation from system to system) spent on I/O *and* networking capability.
  - Users: Compute is precious – *Save Everything!*
- Data is typically analyzed after the simulation is done, if at all.
  - Little in-situ analysis.
- Data typically dumped non-optimally.
  - Do what is easy for the user to code or what made sense to them. Ex) N→1 Writes
  - No standardization of I/O formats – a moving target.

The Bandwidth Gap:

Networks
Disk I/O Bandwidth
Displays (Pixel Count)

Computational Power
Data Set Size (Bytes)

Orders of Magnitude Mismatch!

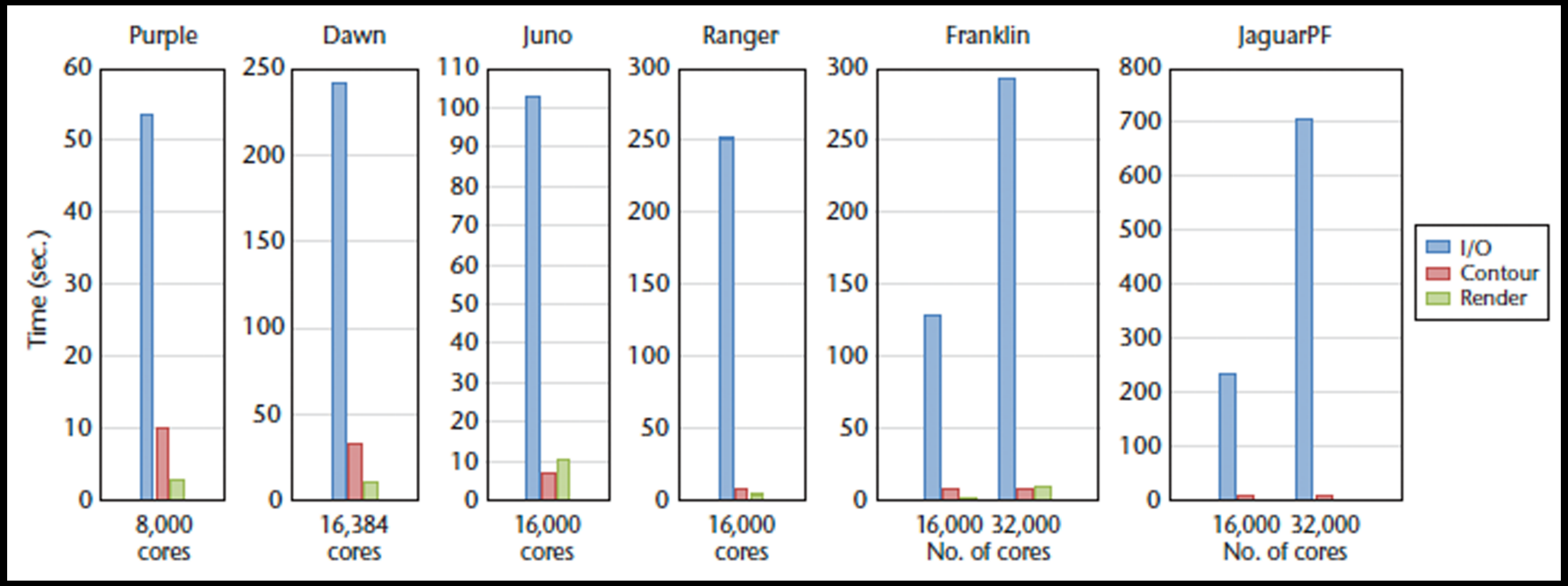
What Does Analysis I/O Look Like?

- It is not just about Restarts!
  - All simulations result in some form of analysis.
  - Need to read data to perform analysis.
- Reads Dominate
  - Need to get raw data into the pipeline, filters and rendering reduces data volume by orders of magnitude.
  - Final product is ~1% of the size of the original input set.
  - Final data products are sent directly to display or easily saved (images are small in comparison to data).
- Want to amortize I/O cost
  - Use same tool to perform visualization and analysis.
  - Visualization drives science question which drives another visualization.

Analysis I/O: More Than Simple Reads

- Bulk Reads.
  - Read in entire data set or slab of variables.
  - Sort and extract in memory (filter's job).
- Reads ideally have a deadline.
  - Want to maintain interactivity with users (or finish batch job quick).
  - Not have your users go for coffee with every operation.
  - Ex.) Interface Response in 10 sec., Data Size = 35 GB
    Need: 3.5 GB/sec (28 Gb/s) from disk to node.
- Reads are synchronized between nodes (every node works in concert with the others on a given time step).
  - Large waves of read requests from large numbers of nodes at the same time.

And it is SLOW!



Core-collapse supernova simulation from the CHIMERA code – @ 3.5 million cells
These results show that, although there is variation across the supercomputers, I/O is the slowest phase by one to two orders of magnitude.

– *Courtesy of Childs, et. al.: Extreme Scaling of Production Visualization Software on Diverse Architectures*

Improving Visualization Read Performance using Distributed File Systems

Solution: VisIO

VisIO: Premise

- Not enough bandwidth to Parallel File Systems via Networks.
  - Need latencies in seconds NOT minutes.
  - We have already established that:
    - Data sets are getting larger AND
    - Visualization reads in large slabs of variables per time step.
- What if we could cut the network out of the picture?

VisIO: Idea…

- What if we collocated compute resources powerful enough to run visualization algorithms with storage?
  - ….. Distributed File Systems!
- Setup a visualization cluster with integrated storage on each node.
  - ….. Data Intensive Supercomputing (DISC)!
- Now we can have each process independently load a portion of the data set for the given time step from the local storage bus (SAS, SATA, etc).
  - No sharing with other nodes or other clusters!

VisIO: How?

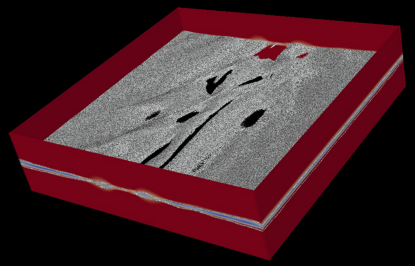
- Use ParaView & Hadoop Distributed File System as Proof of Concept.
  - Rewrite the reader subsystem in ParaView to use libHDFS commands (need C++ compatibility here).
  - Translate libHDFS calls to/from C++ istreams to use existing parser code.
- But how do we know we will only need the data that is local?

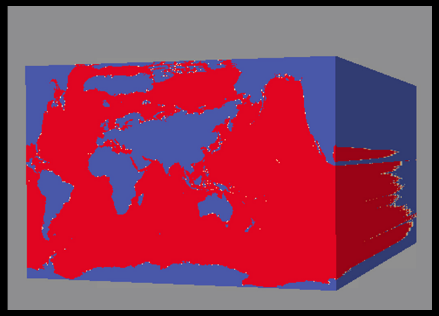
VisIO: Scheduling

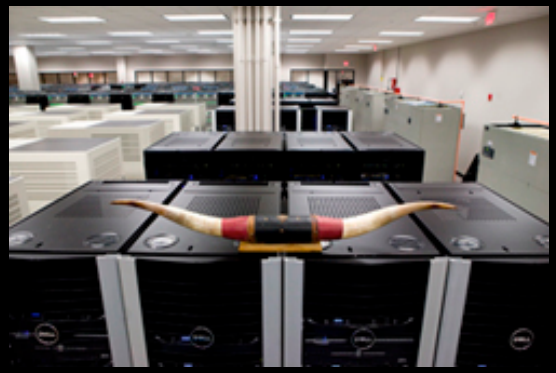
- ParaView's native file format is a binary file with an XML header that specifies the whole and sub extents of the data.
- Query NameNode for location of each piece file.
- Use Stable Marriage Algorithm to greedily produce a schedule across the nodes in the cluster.
  - Schedule scarcest file first.
- When ParaView asks for a subextent – make sure that node who is scheduled for that file takes the task.
  - No internode data transfers if an ideal schedule is produced! ← We have cut the network out of the picture!

VisIO: Prove It!

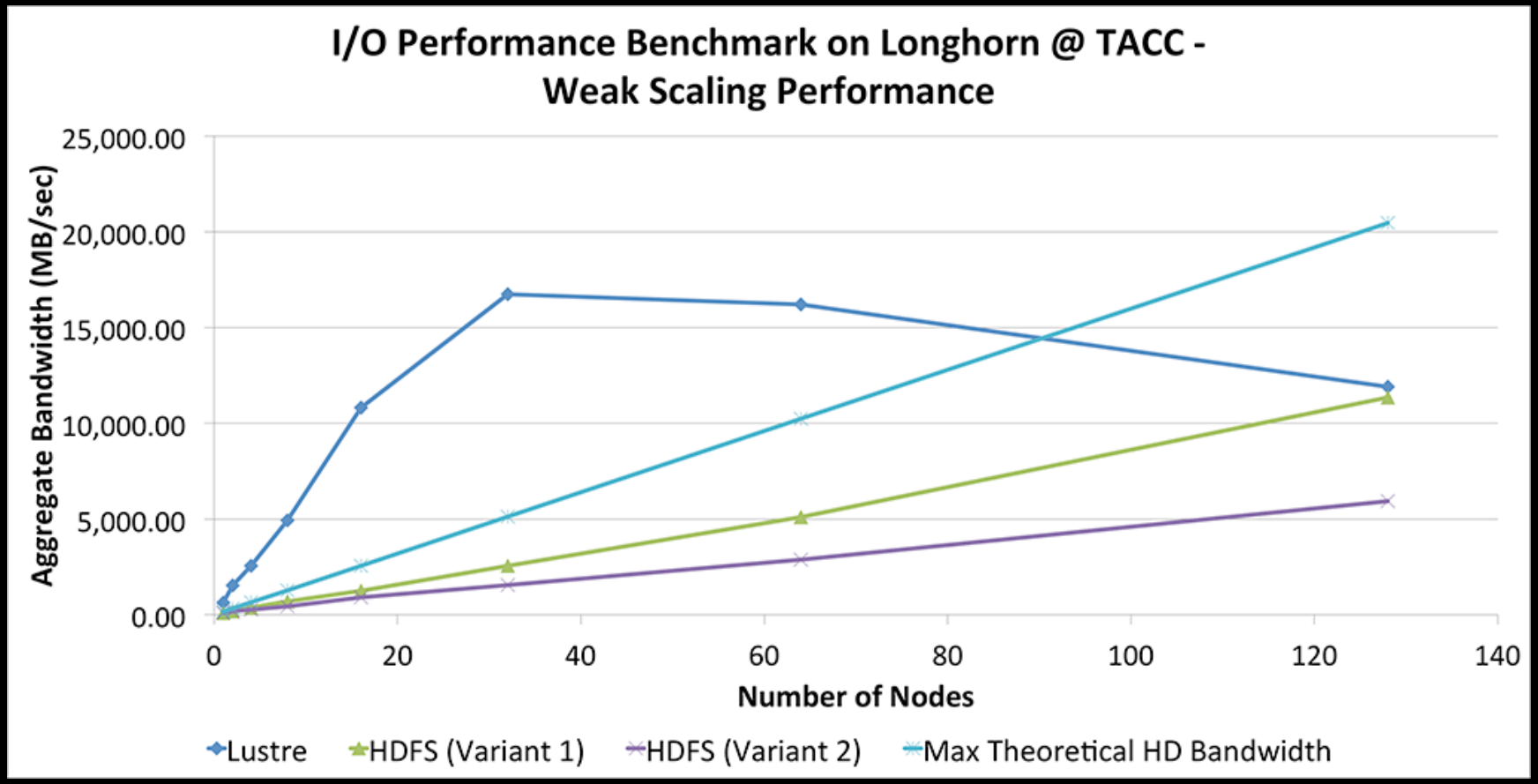
- Tested with two real datasets
  - VPIC
  - Ocean Salinity (from POP)
- ParaView 3.8.0 & Hadoop 0.20.2
- TACC Longhorn Visualization Cluster
  - 256 Total Nodes
    - Dell PowerEdge R610 (240 nodes)
    - Dell PowerEdge R710 (16 nodes)
  - 2 Intel Nehalem Quad Core Processors @ 2.53 GHz.
  - 48 GB RAM (R610) / 144 GB RAM (R710)
  - *73 GB 15K RPM SAS Hard Drive (1 per node)*
  - 4x QDR Mellanox InfiniBand Fabric
  - 2 NVidia Quadro FX 5800 GPUs
  - Lustre Parallel File System (210 TB)
  - CentOS 5.4
- TeraGrid Allocation: TG-ASC100033
  - 150,000 SU (2010-2011)
  - Renewal In Review for 2011-2012
  - Co-PIs: Mitchell, Ahrens, Geveci



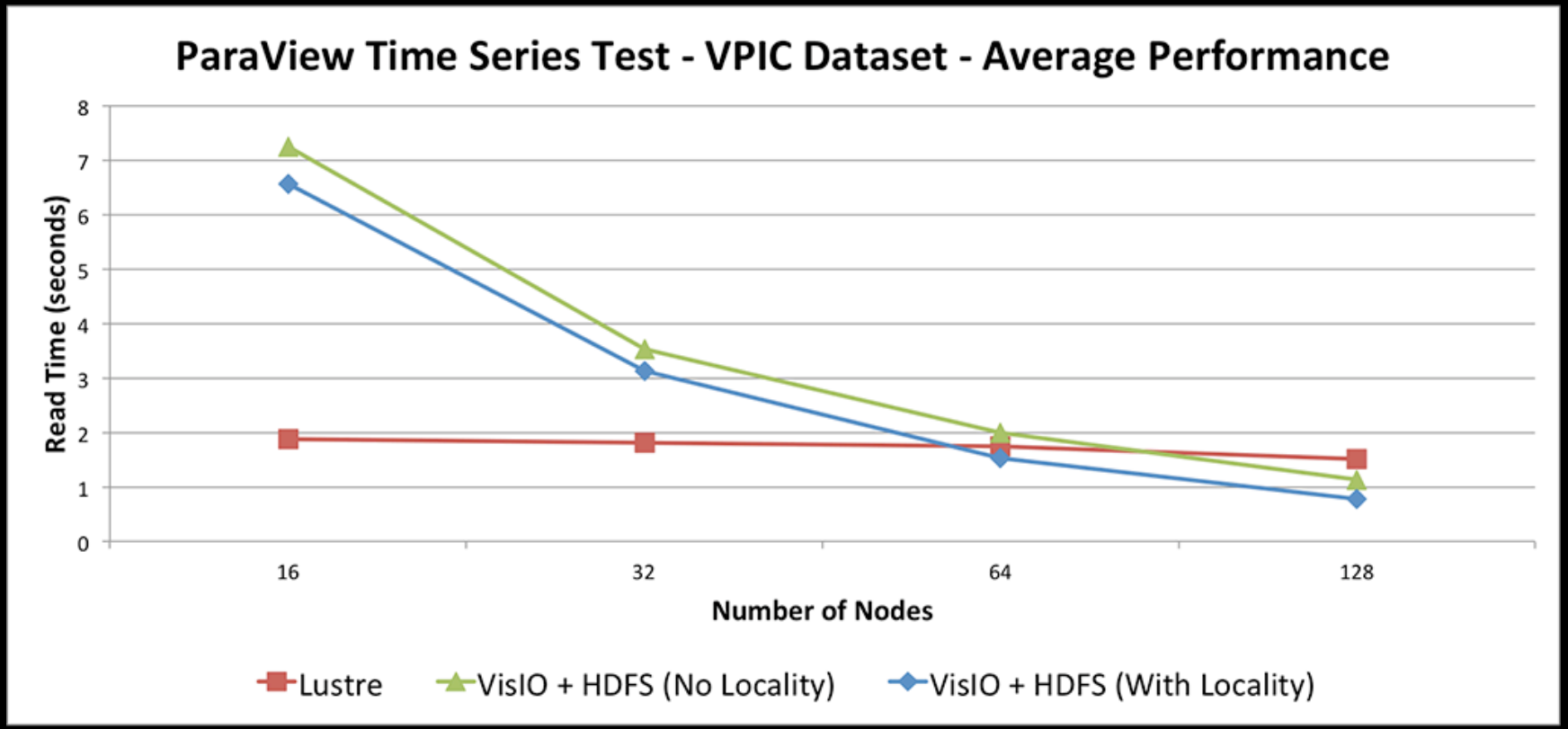




VisIO: Results

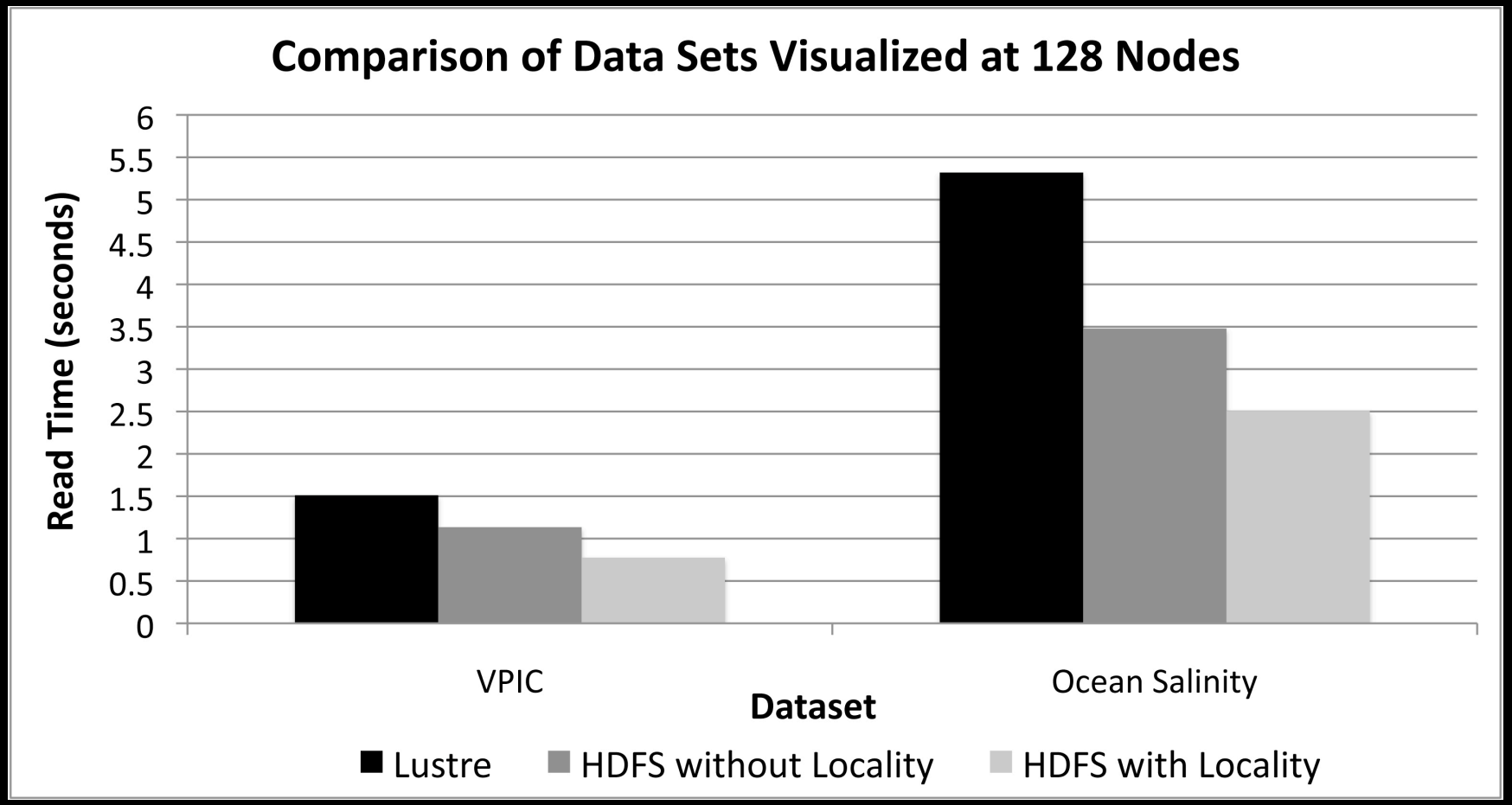


VisIO: Results



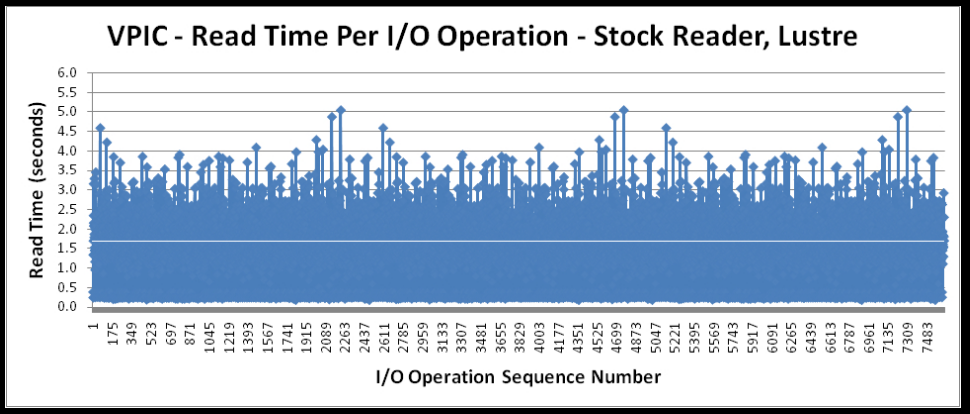
64.38% Improvement at 128 nodes when using HDFS with Locality Scheduling.

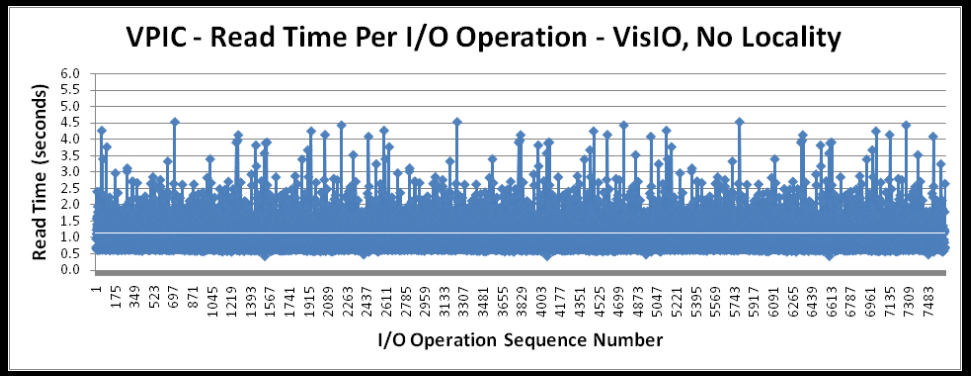
VisIO: Results

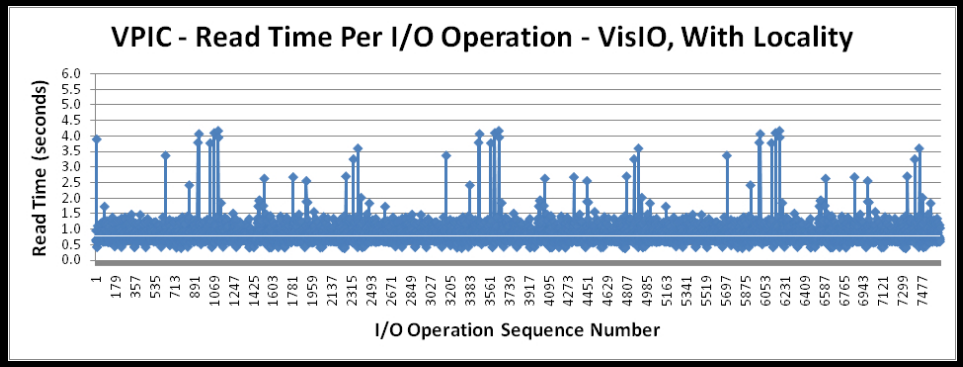


51.43% *Improvement at 128 nodes when using HDFS with Locality Scheduling for Ocean Salinity Data*

VisIO: Results







Notice the decrease in standard deviation around the marked mean.

→Improved consistency in read performance.

→Spikes in read time == remote reads.

VisIO: Conclusions

- *This is a real implementation of Data Intensive Supercomputing concepts.*
- It is possible to interface visualization applications with Distributed File Systems.
- The removal of the network from the data path has yielded impressive speedups in read time.
- Scheduling processes to co-locate with stored data is key to maximum performance.
- A scaled approach with this method can yield interactive visualization of Petascale and future Exascale datasets.

Rethinking I/O access patterns for Visualization

SOLUTION: CLIMATE VISUALIZATION WITH THE PARALLEL OCEAN PROJECT

I/O for POP: Work Flow

- N to 1 write for each timestep
  - 1 file/timestep
- 1 file to N nodes when reading for analysis

- *Circle of Influence allows us to change how we read the data not how we write the data.*
- *There is a disconnect in the workflow between what the simulation does and what the analysis tool is optimized for.*

I/O for POP: Observations

- Entire data set must be read into memory
- Visualization community has done a lot of work on parallel algorithms – for idealized distribution of data in memory.
- Typically use a data parallel processing method.
  - Works well for the visualization and analysis algorithms themselves.
  - Simulation output is not always aligned to this pattern.
- We can read the data faster if we access it in a way that complements the analysis algorithms.

I/O for POP: A Data Set of Interest

- 3600 x 2400 x 42 spatial resolution
- 365 time steps
- 1 floating point scalar field
- 1.4 GB per time step
- single file per time step
- 529 GB for entire data set

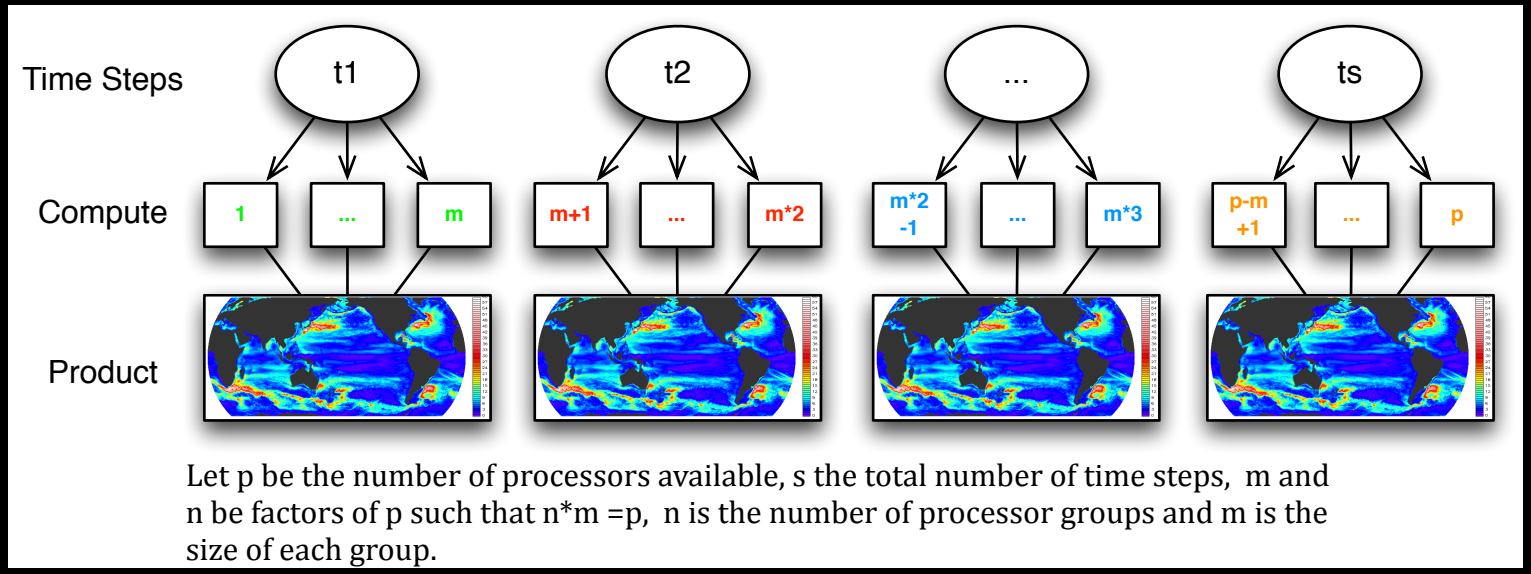
I/O for POP: Traditional Processing

- The big 3 parallel visualization front ends
  - For each time step
    - Each of n processors read 1/n of time step
      - produce geometry
      - Render
    - Composite image

- Processes 1 time step at a time
- Effectively uses parallel rendering and compositing

UVCDAT: Use Case 1

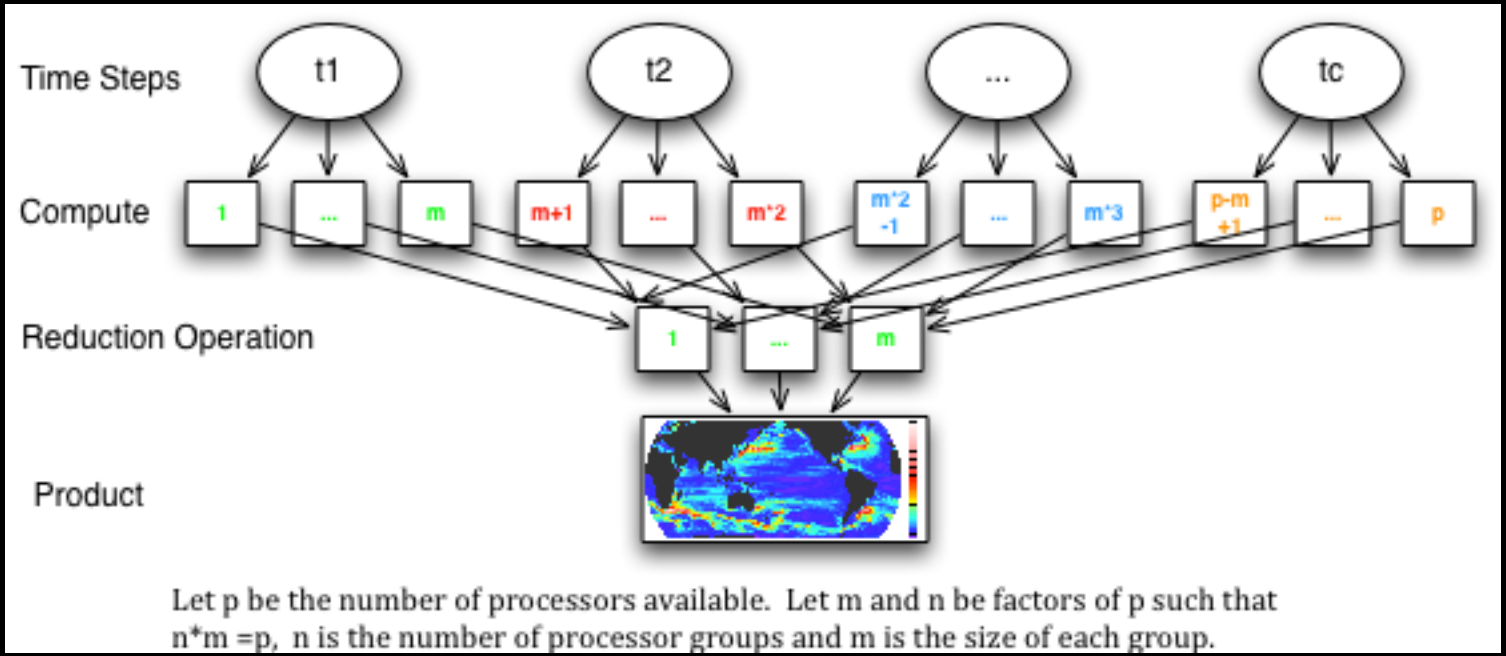
High spatial resolution, time and space parallel, image sequence production



Let p be the number of processors available, s the total number of time steps, m and n be factors of p such that n*m =p, n is the number of processor groups and m is the size of each group.

UVCDAT: Use Case 2

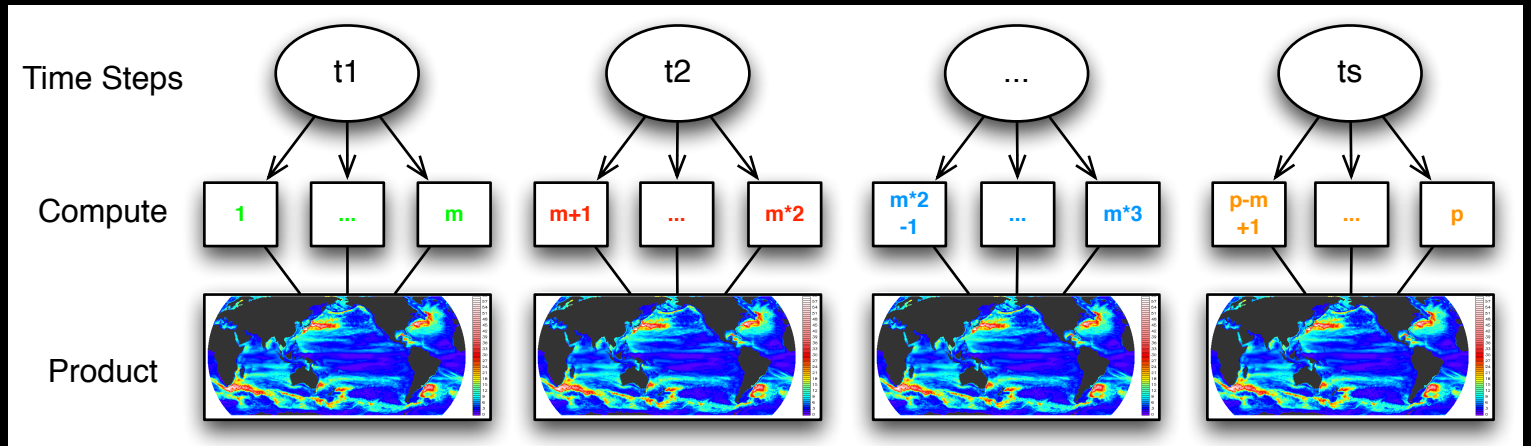
High spatial resolution, time and space parallel, time average



Let p be the number of processors available. Let m and n be factors of p such that n*m =p, n is the number of processor groups and m is the size of each group.

I/O for POP: New Processing

- Divide parallel resources into time compartments
- Allows balancing of Amdahl's law between reads and the rest of the pipeline



Let p be the number of processors available, s the total number of time steps, m and n be factors of p such that n*m =p, n is the number of processor groups and m is the size of each group.

I/O for POP: Results

- We are seeing better than 10x increases in total processing time for a canonical use of the first use-case.
- We're investigating visualization and analysis algorithms to process data distributions as they stand after reading the data efficiently

Questions?

Project Supported By:

U.S. DEPARTMENT OF ENERGY

XSEDE
Extreme Science and Engineering Discovery Environment